Hochschule für Technik
und Wirtschaft Berlin
University of Applied Sciences

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT

BELEGARBEIT IM FACH INFORMATION SYSTEMS

Informationsextraktion mit Apache OpenNLP

Verfasser:
Sergej MEISTER
Matrikelnummer:
s0521159

Dozent:
Prof. Dr. Christian HERTA

2. Oktober 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

Abstrakt

Die Belegarbeit besteht aus zwei Teilen einen praktischen und einen theoretischen Teil. Im praktischen Teil wurde eine Anwendung entwickelt, die spezifische Job-Daten aus HTML-Dokumenten extrahiert und darstellt. Der schriftliche Teil beschreibt die Vorgehensweise, die eingesetzte Technologien und Problemen, die erst in der Entwicklungsphase aufgetreten sind. Das entwickelte Programm wird in zwei Projekte unterteilt: ***civis-opennlp*** und ***intellijob***. Beide Projekte können von GitHub heruntergeladen werden.

civis-opennlp: `https://github.com/SergejMeister/civis-tools.git`
Beschreibung: Enthält die Logik um Job-Metadaten aus HTML-Dokumenten zu extrahieren.

intellijob: `https://github.com/SergejMeister/intellijob.git`
Beschreibung: Verantwortlich für Benutzeraktion, Datendarstellung und Datenanalyse.

1 Einleitung

Auf der Suche nach einer Arbeit muss ein Arbeitssuchender viele Arbeitsangebote auf verschiedenen Job-Portals wie StepStone oder Monster lesen und analysieren. Wenn das Arbeitsangebot den erwartenden Vorstellungen und vorhandenen Qualifikationen entspricht, wird eine Bewerbung an den Arbeitgeber gesendet. Sowohl bei der Analyse eines Arbeitsangebots als auch beim Schreiben einer Bewerbung beschäftigt sich der Arbeitssuchender mit der Informationen, die er aus dem Arbeitsangebot selbst extrahieren muss. Zum Beispiel für die Entscheidung, ob die Stelle den gewünschten Anforderungen entspricht, sind die Daten wichtig, wie Berufsname, Firmenname und Qualifikation. Für das Bewerbungsschreiben werden noch weitere Daten benötigt, wie Kontaktperson, Mailadresse, Firmenhomepage und Firmenadresse. Diese Informationen muss der Arbeitssuchender ständig selbst in einem Arbeitsangebot finden und interpretieren. Im Rahmen dieser Semesterarbeit wird ein System entwickelt, das diese Routine-Arbeit automatisiert und einige Informationen automatisch aus dem Arbeitsangebot extrahiert.
Da das Thema *Informationsextraktion* der Kernstück dieser Arbeit ist, muss es definiert werden, was genau unter diesem Begriff zu verstehen ist. Der erste Satz eines Wikipedia-Artikels *Informationsextraktion* sagt:

> *„Unter Informationsextraktion (engl. Information Extraction, IE) versteht man die ingenieursmäßige Anwendung von Verfahren aus der praktischen Informatik, der künstlichen Intelligenz und der Computerlinguistik auf das Problem der automatischen maschinellen Verarbeitung von unstrukturierter Information mit dem Ziel, Wissen bezüglich einer im Vorhinein definierten Domäne zu gewinnen.“*[1]

Ja, einige Begriffserklärung müssen auch geklärt werden. Also, Informationsextraktion ist ein Verfahren, um strukturierte Informationen aus den unstrukturierten Informationen zu gewinnen. Bezüglich auf die Belegarbeit bedeutet es die Informationen wie Kontaktperson, Adresse, Email etc. aus dem unstrukturierten in HTML-Form erfassten Text eines Arbeitsangebots zu extrahieren und in einem Datenbank strukturiert abzuspeichern. Die strukturiert abgespeicherte Informationen können viel leichter dargestellt und analysiert werden. Wenn die Informationen leicht analysiert werden können, kann auch die Entscheidung viel schneller getroffen werden. Alles zusammengefasst lässt sich der Begriff Informationsextraktion wie folgt definieren: Informationsextraktion ist ein Verfahren, um die Daten, die für eine Entscheidung relevant sind, automatisch aus dem Volltext zu extrahieren. Dabei werden unterschiedliche Technologien eingesetzt.

[1] Wikipedia, *Informationsextraktion*

2 Verwendete Technologien

2.1 Reguläre Ausdruck

Einige Informationen in einem unstrukturierten Volltext weisen doch eine bestimmte Struktur nach. Ein gutes Beispiel ist eine Email-Adresse z.B. *sergejmeister@web.de* . Die Struktur von Email-Adresse ist weltweit eindeutig und besteht aus ***name@domain***. Obwohl ***name*** und ***domain*** sehr unterschiedlich geschrieben werden könnten, dadurch dass kein Leerzeichen vorkommen darf, ist eine Mail-Adresse immer ein strukturiertes Wort im Volltext. Solche Daten können mit einem Regulären Ausdruck eindeutig bestimmt und extrahiert werden.

Mail-Pattern: *[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+.[a-zA-Z0-9-.]+*

Weitere Daten, die auch mit einem Regulären Ausdruck leicht gefunden werden können, sind URLs.

WWW-Pattern: *www.[\w\d.:#@%/;$()~_?+-=&]**

HTTP-Pattern: *http.[\w\d.:#@%/;$()~_?+-=&]**

HTTPS-Pattern: *https.[\w\d.:#@%/;$()~_?+-=&]**

Reguläre Ausdrücke eignen sich gut um die Informationen mit einer festen Struktur zu extrahieren. Allerdings sind die meisten Informationen in einem Volltext eines Stellenangebots entweder unstrukturiert oder sehr komplex, um alle mögliche Wortbildungen mit dem Regulären Ausdruck abzudecken. Im Rahmen dieser Belegarbeit muss neben Email und Firmenhomepage auch Firmenadresse aus dem Arbeitsangebot extrahiert werden. Da eine Adresse in Deutschland meistens aus **Straße**, **Hausnummer**, **Postleitzahl** und **Stadt** besteht, kann auf den ersten Blick eine gewisse Struktur nachvollziehen werden. In Wirklichkeit kann die Adresse ziemlich unterschiedlich dargestellt werden.

- Kurfürstenstraße 13 13456 Berlin
- Brandenburger Tor 17 D-13456 Berlin
- Straße des 17. Juni 17a 13456 Berlin

- Adalbert-Stifter-Straße 1 D-45678 Frankfurt am Main
- Alle der Kosmonauten 26a D-32451 Bad Kreuznach

Daraus wird ersichtlich, dass nur **Postleitzahl** und **Hausnummer** über eine gewisse Struktur verfügen. Die restliche Informationen wie **Straße** und **Stadt** sind unstrukturiert und es ist sehr schwierig allein mit Regulären Ausdrücke diese Daten zu extrahieren. Es gibt eine Informatik Disziplin namens Natural Language Processing kurz NLP, die sich genau mit der Verarbeitung natürlichen Sprachen auseinandersetzt.[2]

2.2 Natural Language Processing

Natural Language Processing (deutsch maschinelle Verarbeitung natürlicher Sprache) beschäftigt sich sowohl mit geschriebener (Text) als auch mit gesprochener Sprache. Dabei wird das Wissen über Morphologie, Syntax und Semantik aus der natürlichen Sprache berücksichtigt. Das Ziel ist nicht nur die Bedeutung eines einzelnen Wortes, sondern viel mehr sein Zusammenhang mit anderen Wörtern, ganzen Sätzen oder Sachverhalten zu erkennen. Dieses Verfahren nennt sich Muster-Erkennung und basiert auf künstlich erzeugte Erfahrungen so genannten Trainingsdaten. *"Der große Vorteil an dieser Methode besteht darin, dass die Computer immer besser werden, je mehr Daten sie erhalten. Ein gutes Beispiel hierfür ist die Übersetzungsfunktion von Google. Zu Beginn wurde das Projekt noch vielfach belächelt. Heute ist das Programm in der Lage, viele verschiedene Texte und selbst das gesprochene Wort einigermaßen flüssig zu übersetzen."* [3] Es gibt mehrere Frameworks, die sich mit NLP beschäftigen und ein davon ist **Apache OpenNLP**.

2.3 Apache OpenNLP

Apache OpenNLP ist ein Open Source Produkt, das einige Werkzeuge für maschinelle Verarbeitung natürlicher Sprache zusammenstellt. Das Programm ist in der Programmiersprache JAVA entwickelt und kann als Maven-Dependency in ein anderes Projekt eingebunden werden. Der Produkthersteller bietet auch alternativ eine Kommandoschnittstelle, um die bereits erstellte Modelle und Komponente nutzen zu können. Die meisten Komponente basieren sich auf Methoden der Maximale-Entropie. Das Verfahren ist sehr komplex und wird deswegen im nächsten Kapitel näher erläutert. Die Tabelle 1 stellt OpenNLP-Komponenten dar, die seit Version 1.5.0 vorhanden sind. Jede Komponente muss mit einem lokalisierten Model initialisiert und

[2] Freiknecht, „Big Data in der Praxis“
[3] OnPageWiki, *Natural Language Processing*

Tabelle 1: OpenNLP 1.6.0 Funktionalität

Komponenten	**Beschreibung**
SentenceDetector	Teilt Text in einzelne Sätze
Tokenizer	Teilt Satz in einzelne Worte
Named Entity Recognition	Erkennt und klassifiziert Bestandteile im Text
Part-Of-Speech tagging	Das Zuweisen von Markierungen zu einzelnen Einheiten (Wortart-Annotierung)
Chunker	Teilt Text in syntaktisch korrelierten Teile von Wörtern, wie Nomen Gruppen
Parser	Verwendet Tokenizer und Chunker
Coreference Resolution	Bezugnahme auf dieselbe Entität
Document Classification	Text - Klassifizierung

ausgeführt werden. Einige Modelle werden von OpenNLP bereitgestellt und können von der Herstellerseite heruntergeladen werden.[4] Das vorhandene Person-Modell *„en-ner-person.bin“* ist nur für englische und spanische Sprache verfügbar und kann leider nicht effektiv zum Extraktion von Kontaktperson verwendet werden. Deswegen muss eigenes Modell entwickelt werden. Jedes Modell muss mit lokalisierten Daten trainiert werden. Die empfohlene Anzahl von Trainingsdaten laut Produkthersteller liegt bei 15.000 Datensätzen. Die Entwicklung von Kontaktperson-Modell und Firmenadresse-Modell wird noch detailliert in weiteren Kapiteln beschrieben.
OpenNLP wird ständig weiterentwickelt und verfügt über eine gute Dokumentation [5], die die Einarbeitungszeit wesentlich erleichtert. Obwohl am 13 Juli 2015 die neueste Version 1.6.0 released wurde, wird in der Belegarbeit noch die alte Version 1.5.3 verwendet.

[4] `http://opennlp.sourceforge.net/models-1.5/`
[5] `https://opennlp.apache.org/documentation/1.6.0/manual/opennlp.html`

3 Maximum Entropie

Die Methoden Maximale Entropie werden für die Datenklassifizierung verwenden. Der Begriff Entropie kommt ursprünglich aus Physik und bezeichnet in Informationstheorie einen mittleren Wert der Auftrittswahrscheinlichkeit (Informationsgehalt). Falls Ergebnis genau ermittelt werden kann, ist die Entropie gleich 0. Je größer die Entropie ist, desto schwieriger ist es die Wahrscheinlichkeit und somit auch die Zugehörigkeit zu einer Klasse zu erfassen. Wieso heißt dieses Verfahren Maximum Entropie, wenn die Klassifizierung mit dem maximalen Wert der Entropie kaum bestimmt werden kann? Die Antwort liegt in der Wahrscheinlichkeitsverteilung. Das Bild unten zeigt die Auftrittswahrscheinlichkeit und Entropie für zwei möglichen Ergebnissen.

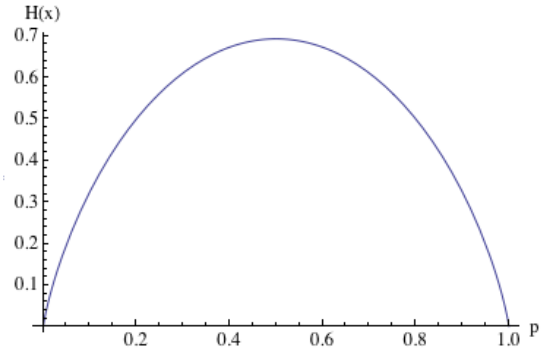


Abbildung 1: Auftrittswahrscheinlichkeit und Entropie

Die Maximale Entropie auf dem Bild ist gleich **0.5** und die Wahrscheinlichkeit, dass ein **0** oder **1** in dem Punkt auftritt, ist gleich. Man spricht deswegen über eine Gleichwahrscheinlichkeit. Wird aber die Ergebnismenge erhöht, so ändert sich auch die maximale Entropie und zwar der Wert wird kleiner.[6] Der Grundprinzip der Maximale Entropie lautet

> *„Ist auf der Grundlage unzureichender Information aus einer Vielzahl von Wahrscheinlichkeitsverteilungen eine Verteilung auszuwählen, dann ist genau diejenige zu nehmen, welche die größte Entropie besitzt und mit der gesamten verfügbaren Information übereinstimmt.“*[7]

Dieses Prinzip basiert auf einem bekannten Grenzverteilungssatz für die empirische Entropie,

$$S_N = -\sum_{i=1}^{n} p_i ln(p_i)$$

[6] Bazenov, *Classification with Maximum Entropy*

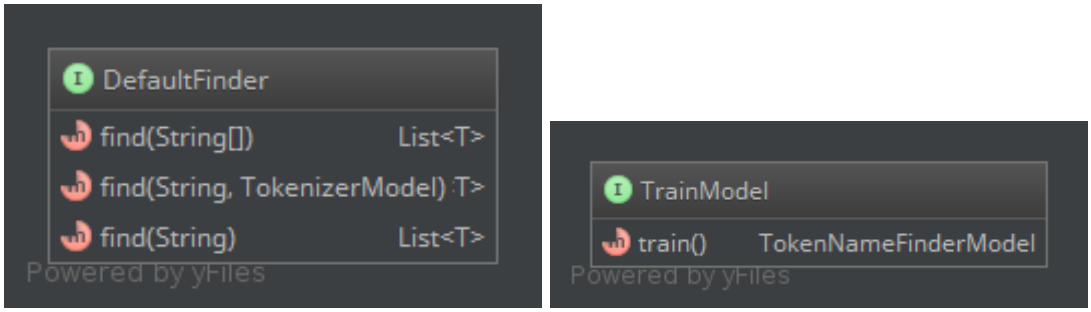
[7] Reiter, „Bildverschärfung durch Lösung der Fredholmschen Integralgleichung 1. Art mittels der Maximum-Entropie-Methode mit astronomischen Anwendungen“

, wo S Entropie mit $S(\pi_1, ..., \pi_n) \rightarrow MAX$ und p_i Auftrittswahrscheinlichkeit ist.[8] Dieses Verfahren ist genau der Grund, wieso Produkthersteller von OpenNLP für die Erstellung eigener Modelle empfehlt, min. 15.000 Trainingsdatensätzen bereitzustellen. Denn durch große Anzahl von Trainingsdaten wird Auftrittswahrscheinlichkeit sehr gut verteilt und Maximale Entropie wird sehr gering.
Falls die Anzahl von Trainingsdaten nicht ausreichend groß ist oder die gelieferte Ergebnisse den Erwartungen nicht entsprechen, dann können Nebenbedingungen, so genannten *Features*, eingesetzt werden. Die Nebenbedingungen sind zusätzliche Informationen, die das Verfahren Maximale Entropie wesentlich verbessern. Eine gute Nebenbedingung für deutsche Sprache ist zum Beispiel, ein großer Buchstabe am Wortanfang, was ziemlich eindeutig auf ein Nomen hinweist.

[8] Duller, „Die Maximum-Entropie-Methode zur Bestimmung von Mischanteilen“

4 Models

In diesem Kapitel wird beschrieben, wie eigene Modelle mit Hilfe von Apache OpenNLP Framework erstellt werden können. Jedes Modell beinhaltet zwei Algorithmen *Trainingslogik* und die Logik zur Ermittlung des besten Treffens, was weiter in der Belegarbeit als *Finderlogik* bezeichnet wird. So werden zwei Interfaces DefaultFinder.java und TrainModel.java erstellt.



(a) DefaultFinder.java (b) TrainModel.java

Abbildung 2: Interfaces - DefaultFinder und TrainModel

Wobei die Logik um Modell zu trainieren, ist für alle Modells gleich. Deswegen wird es in einem abstrakten Klasse **BaseModel.java** implementiert. Die wichtigste Trainingsparameter sind ***Algorithm***, mit welchem das Model trainiert werden muss, ***Iteration***, die Anzahl, wie viel Mal der Algorithm durchlaufen soll und ***cutoff***, die Anzahl, wie viel Mal ein Feature auftreten muss, damit es von dem Algorithm berücksichtigt wird. Die Standartwerte für diese Parameter sind ***Algorithm*** MAXENT, ***Iteration*** 100 und ***cutoff*** 5. Die weitere Trainingsparameter, die auch benötigt werden, sind Trainingsdaten, Modellname, Modellsprache und optional Features.

```
TrainingParameters trainingParameters = trainConfigData
   .getTrainingParameters();
NameFinderEventStream ss = new NameFinderEventStream(
   trainConfigData.getSamples(), trainConfigData.getType(),
   new DefaultNameContextGenerator(trainConfigData.getFeatureGenerator()));
AbstractModel nameFinderModel = TrainUtil.train(
   ss, trainingParameters.getSettings(), null);
return new TokenNameFinderModel(trainConfigData.getLanguageCode(),
    nameFinderModel, null, null);
```

Wenn Modellname, im Code getType(), gleich ***„contact-person "*** und Sprache gleich ***„de"*** sind, wird ein Maxent-Modell ***„de-contact-person.bin"*** erstellt. Vor der Verwendung des *TrainLogik* oder *FinderLogik* wird der Text mit dem Tokenizer und Sentence-Detector auf einzelne Sätze und Worte zerlegt.

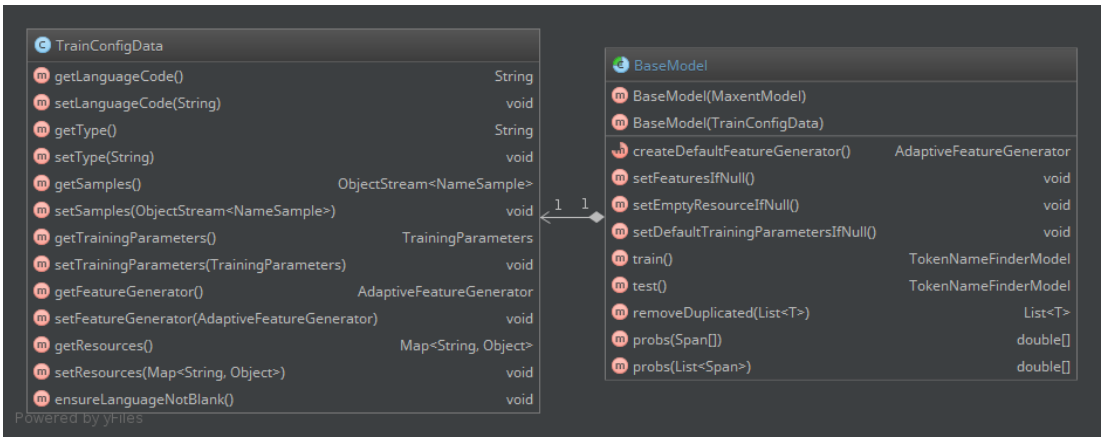


Abbildung 3: Abstrakte Klasse BaseModel.java für alle Modelle

4.1 Contact Person Model

Die Trainingsdaten für das Modell **„ContactPerson“** liegen in einem Text-File **contact-train.txt**. Die erwartende *NameEntity* muss mit einem START- und END-Tag eindeutig markiert werden. Es ist auch zu achten, dass Tags und *NameEntity* durch ein Leerzeichen getrennt werden müssen. Ein Trainingssatz für das Modell **„ContactPerson“** kann wie folgt aussehen:

> Wenn Sie sich angesprochen fühlen steht Ihnen unser Berater, **<START:salutation>** Herr **<END>** **<START:person>** Klaus Mustermann **<END>** , für erste Informationen zur Verfügung.

In dem Beispiel oben werden zwei Markierungen, eine für Anrede und die andere für Person, verwendet. Aus persönlicher Beobachtung wird eine bessere Ergebnismenge erreicht, wenn Anrede auch markiert wird.
Insgesamt verfügt ContactPerson-Modell über 1822 Datensätzen. Die Zahl liegt natürlich deutlich unter die empfohlene Grenze 15.000. Deswegen wurden sinnvolle Features eingesetzt, die die Ergebnismenge wesentlich verbessert haben.

FirstCapitalLetterFeature

Vorname und Nachname beginnen immer mit dem großen Buchstabe. Deswegen kann es sinnvoll sein, alle solche Worte mit dem Prefix ***„flup“*** zu annotieren.

> [(flup=wenn) (flup=sie) (sich) (angesprochen) (fühlen) (steht) (flup=ihnen) (unser) (flup=berater), (flup=herr) (flup=klaus) (flup=mustermann) (,) (für) (erste) (flup=informationen) (zur) (flup=verfügung) (.)]

ContactPersonFeatureGenerator

Vor- und Nachname stehen nebeneinander und bestehen immer nur aus Buchstaben, keine Zahlen und kein Sonderzeichen außer Minuszeichen (-). **ContactPersonFeatureGenerator** prüft mit einem Regulären Ausdruck [*^[A-Z][a-z][a-z-]**] , ob das Wort dem Muster entspricht. Fall ja, wird es mit dem Prefix ***„np“*** annotiert. Außerdem prüft dieses Feature, ob das nächste Wort auch dem Muster entspricht und beim positiven Ergebnis annotiert das Wort mit dem Prefix ***„nnp“***.

FirstNameFeatureGenerator

Dieses Feature bezieht sich auf eine Liste mit 18.484 deutschen Vornamen. Das bedeutet, wenn ein Wort gleich dem Wort in der Liste ist, dann kann es der Vorname sein. Das Wort wird mit dem Prefix ***„fn“*** annotiert.

Sonderfall

Dank Features konnte die Ergebnismenge verbessert werden, jedoch nicht wie erwartet. Natürlich mit steigenden Anzahl von Trainingsdaten wird das Modell immer besser und besser. Das kann aber eine gewisse Zeit in Anspruch nehmen. Deswegen wird es versucht, die Kontaktperson auch mit logischen Mitteln zu bestimmen, wenn OpenNLP leere Ergebnis liefert. Dabei wird einfach nach dem Vorname gesucht, die in der Liste von ***FirstNameFeatureGenerator*** drin sind, und sowohl Vorname als auch das nächste Wort als Ergebnis interpretiert.

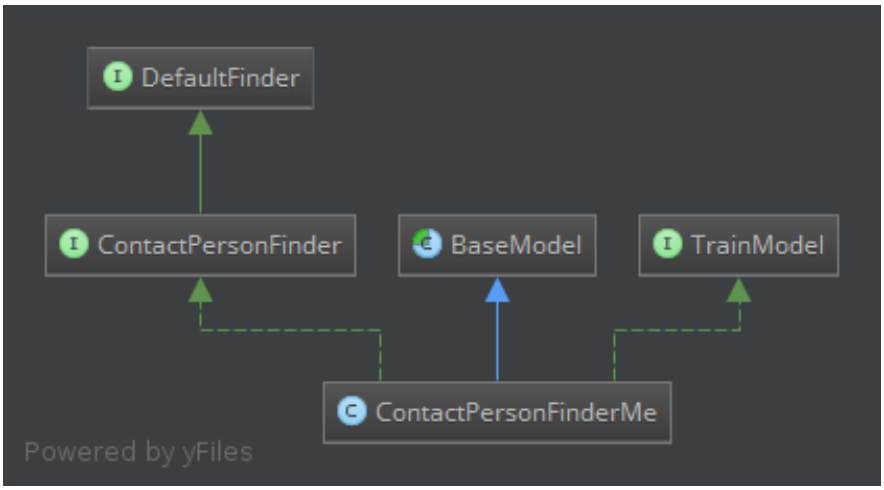


Abbildung 4: Klassendiagramm - Modell - ContactPersonFinderMe

4.2 Address Model

Die Datei *address-train.txt* mit Trainingsdaten für **„AddressModell“** enthält noch weniger Daten als Modell **„ContactPerson“**. Deswegen ist auch die Ergebnismenge in dieser Version noch ziemlich ungenau. Der Trainingssatz wird mit *NamedEntity address* markiert.

> Richten Sie bitte Ihre Bewerbung an Dr. Jürgen Tesla, Firma GmbH, **<START:address>** Rudower Chaussee 29, 12489 Berlin **<END>** oder gerne auch per Mail an jtesla@firma.de.

Genau so wie für das Modell **„ContactPerson“** werden auch für dieses Modell sinnvolle Features entwickelt.

NumberFeature

Die Hausnummer und die Postleitzahl einer Adresse sind meistens Zahlen und alle Zahlen werden mit dem Prefix ***„numb“*** annotiert.

FirstCapitalLetterFeature

Die Straße, die Stadt und das Land fangen immer mit einem großen Buchstaben an. Deswegen kann dieses Feature auch für dieses Modell eingesetzt werden.

StreetNumberFeature

Wie schon erwähnt wurde, besteht die Straßennummer meistens aus Ziffern. Es gibt allerdings einige Ausnahmen wie **„13a“**, **„105-106“** oder **„17/18“**. Dieses Feature prüft mit einem Regulären Ausdruck *[0-9/-]{1,5}[0-9a-zA-Z]$* , ob das Wort dem Muster entspricht und markiert es mit dem Prefix ***„sn“***

AddressFeature

Das letzte Feature prüft, ob eine folge von Tokens dem Adressmuster entspricht. Das heißt, ob das erste Wort mit einem großen Buchstabe beginnt und nur aus Buchstaben und Minuszeichen (-) besteht. Das zweite Token muss dem **StreetNumberFeature** entsprechen. Das dritte Token muss eine Postleitzahl sein und es werden nur

die Ziffern erwartet. Das vierte Token ist die Stadt und genau so wie die Straße darf nur Buchstaben und Minuszeichen (-) haben. Wenn alles zustimmt, dann werden die entsprechende Tokens mit Präfixes markiert: ***„ms“*** die Straße, ***„msn“*** die Straßennummer, ***„mz“*** die Postleitzahl und ***„mc“*** die Stadt.
Dieses Feature ist nicht optimal, denn es berücksichtigt nicht die Straßen und Städte, die aus mehreren getrennten Worten bestehen, wie *Potsdamer Platz* oder *Frankfurt am Main*. Trotzdem ist dieses Feature sinnvoll, weil es die folge von Worten und Zahlen berücksichtigt.

Sonderfall

Genau wie für das Modell **„ContactPerson“** wird es auch versucht, die Adresse mit logischen Operationen zu bestimmen, wenn OpenNLP leere Ergebnis liefert. Dabei wird zuerst eine Liste mit 59171 deutschen Postleitzahlen initialisiert. Jede Postleitzahl ist einer Stadt zugeordnet. Wenn die Postleitzahl und dazu gehörige Stadt in dem Text vorkommt, dann wird das Token vor der möglichen Postleitzahl nach Straßennummer geprüft, entspricht das Token der Straßennummer, wird noch das Token vor der Straßennummer nach Straßenmuster geprüft. Wenn alles erfolgreich erkannt wurde, wird eine Adresse gebildet und als Ergebnis interpretiert. Wenn man die Logik noch einmal genau liest, wird es klar, wie es zu der Ergebnis ***Straße 13 13307 Berlin*** kommen könnte.

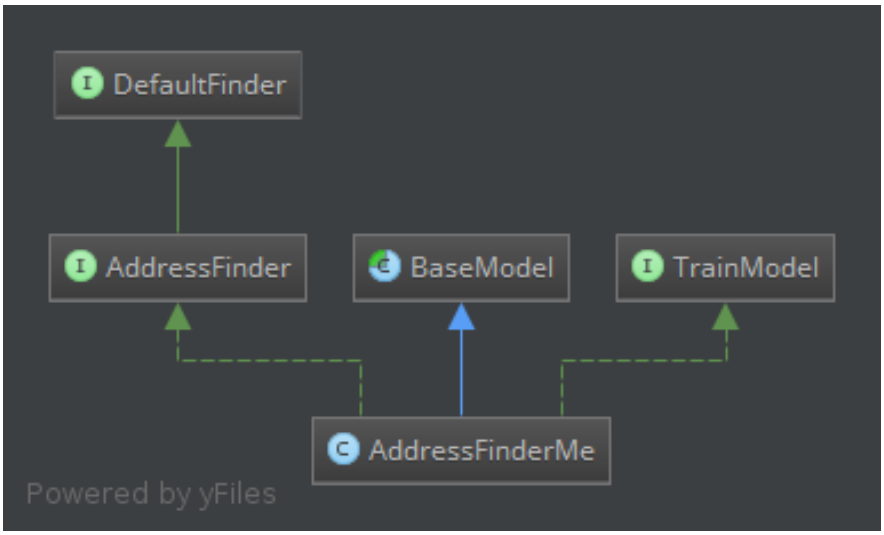


Abbildung 5: Klassendiagramm - Modell - AddressFinderMe

5 Anwendungsdaten

Im Kapitel 4 wurde gezeigt, wie eigene Modelle mit Daten trainiert werden können. In diesem Kapitel geht es darum, zu zeigen, wie die Daten ermittelt, dargestellt und analysiert werden. Für diese Zwecke ist eine Webanwendung namens „**IntelliJob**“ mit Spring und AngularJS Framework entwickelt worden. Die Kommunikation zwischen Back-End (Spring) und Front-End (AngularJS) erfolgt über REST-Services. Alle Daten werden in einen dokumentorientierten Datenbank **MongoDB** gespeichert.

5.1 Datenermittlung

Job-Agent erstellen

Die Anwendungsdaten, die im Rahmen dieser Belegarbeit erfasst werden müssen, sind Arbeitsangebote. Um Arbeitsangebote auf eigene Mail-Adresse zu erhalten, wurden zwei Job-Agenten auf Job-Portals Stepstone und Monster erstellt.

Emails holen

Durch Benutzeraktion werden alle Arbeitsangebote aus dem Email-Postfach gelesen. Dafür muss der Benutzer die Zugangsdaten seines Email-Accounts der Anwendung „**IntelliJob**“ bekannt geben. Die Email-Zugangsdaten werden nur für die Autorisierung verwendet und nie gespeichert. In der ersten Version werden nur zwei Mail-Provider Gmail.com und Rambler.ru unterstützt. Nach der erfolgreiche Speicherung von allen Mails mit Arbeitsangeboten wird ein Zeitstempel des letzten Mail-Abrufs auch in Datenbank gespeichert. Bei der wiederholten Ausführung werden nur Mails abgerufen, deren Empfangsdatum größer als Zeitstempel des letzten Mail-Abrufs ist.

Link zum Arbeitsangebot ermitteln

Die Mails beinhalten Links sowohl zu Arbeitsangeboten als auch zu unterschiedlichen Werbungen. Um die Links von einander zu trennen, wurde ein Strukturmuster entdeckt. Die Links zu den Arbeitsangeboten von Monster-Portal fangen immer mit „**stellenanzeige.monster.de**“ und von Stepstone-Portal mit „**www.stepstone.de/ja.cfm**“ an. Alle Links zu Arbeitsangeboten werden auch in Datenbank gespeichert. Interessanterweise enthalten alle Links die Berufsbezeichnung als Value. Dadurch wird die erste Metainformation, Berufsbezeichnung, erfolgreich ermittelt.

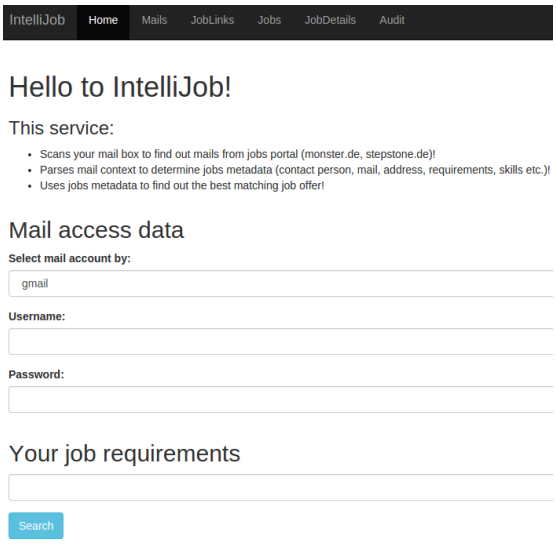


Abbildung 6: Web-Form für Mail-Zugangsdaten

Link zum Arbeitsangebot aufrufen

Als Nächstes werden die Links aufgerufen, um der Inhalt des Arbeitsangebots zu ermitteln und zu speichern. Nachdem das Arbeitsangebot erfolgreich in Datenbank gespeichert wurde, wird auch das entsprechende Link als „**downloaded**“ markiert. Das hat zur Folge, dass die Links mit dem Flag „**downloaded**“ gleich TRUE beim nächsten Mal nicht mehr berücksichtigt werden.

Daten aus dem Arbeitsangebot ermitteln

Der Inhalt des Arbeitsangebot wurde in einem HTML-Form abgespeichert. Um die Daten mit OpenNLP extrahieren zu können, muss der Inhalt zuerst in Plain-Text formatiert werden. Danach werden mit Hilfe von Reguläre Ausdrücke Firmenhomepage und Bewerbungsmail ermittelt. Als Nächstes wird OpenNLP eingesetzt, um die Kontaktperson und Firmenadresse zu bestimmen. Zum Schluss wird das Arbeitsangebot auch mit dem Flag „**Extracted**“ markiert, um der wiederholten Extraktionsprozess zu vermeiden. Der ganze Prozess wird unten in einem Diagramm abgebildet.

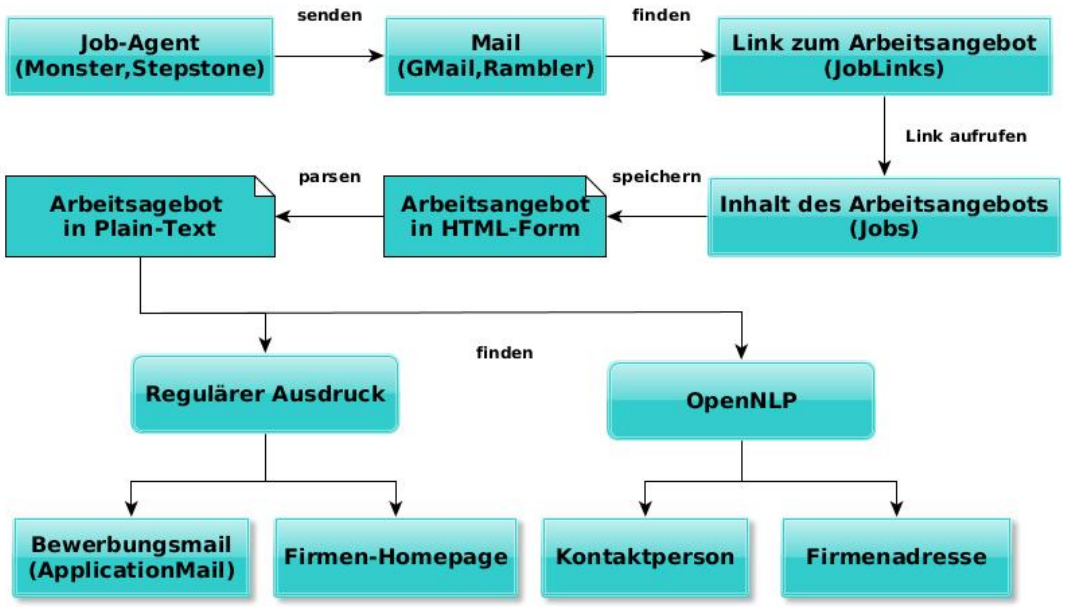


Abbildung 7: Prozessabbildung

5.2 Datendarstellung

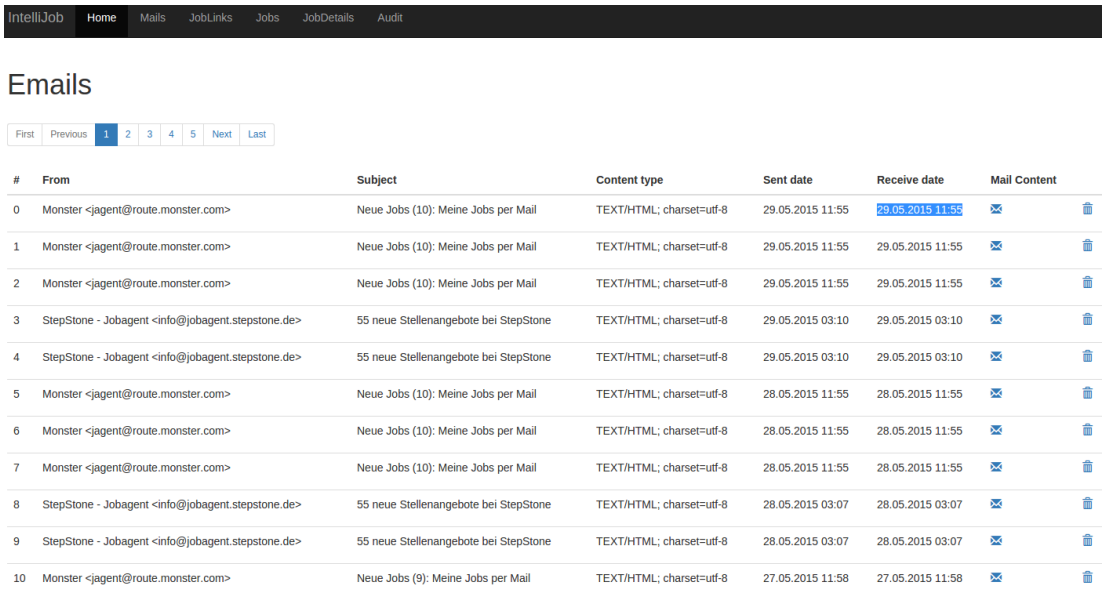
Alle Daten werden einfach in eine tabellarische Form dargestellt. Jeder Prozessschritt hat eigene View mit der entsprechender Tabelle und einige Basisfunktionen wie *Löschen*, *Link aufrufen* und *Details anzeigen*.

Ansicht E-Mails

Die wichtige Daten, die in der Tabelle erfasst werden, sind **Absender**, **Betreff**, **E-Mail Type**, **Versandzeit** und **Empfangszeit**. Alle Tabellenitems sowohl in der Email-Ansicht als auch in allen anderen Ansichten werden nach Empfangszeit absteigend sortiert. Das hat zur Folge, dass aktuellste Mails und Arbeitsangeboten immer zuerst angezeigt werden. Die Icon-Events, **Mail Content** und **Löschen**, sind dazu da, um die Mails anzuzeigen oder zu löschen.

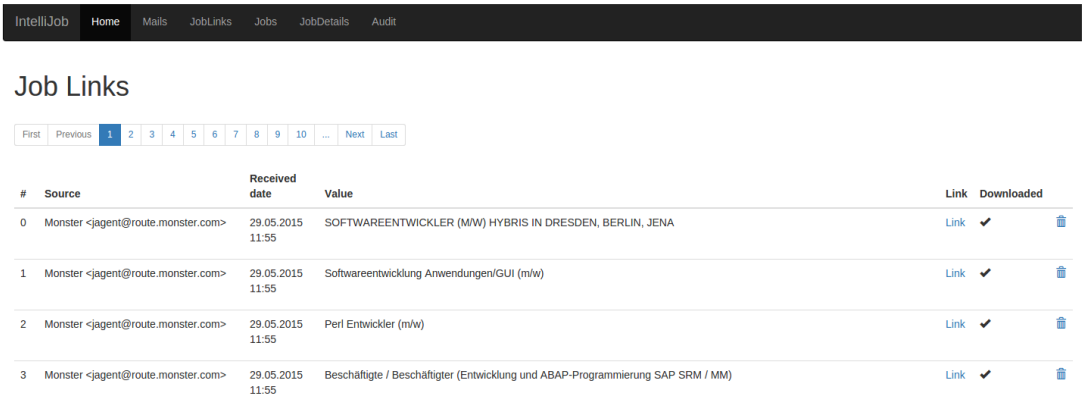
Ansicht Links zu Arbeitsangeboten

In dieser Ansicht werden neben der Empfangszeit nur zwei weitere Informationen angezeigt: **Link-Quelle** (Monster- oder Stepstone-Portal) und **Link-Value**. Wie in dem vorherigen Kapitel schon angedeutet wurde, beinhalten die Links-Value immer eine Berufsbezeichnung. Mit dem Klick auf Link-Button wird der Link aufgerufen. Das Flag



#	From	Subject	Content type	Sent date	Receive date	Mail Content
0	Monster <jagent@route.monster.com>	Neue Jobs (10): Meine Jobs per Mail	TEXT/HTML; charset=utf-8	29.05.2015 11:55	29.05.2015 11:55	
1	Monster <jagent@route.monster.com>	Neue Jobs (10): Meine Jobs per Mail	TEXT/HTML; charset=utf-8	29.05.2015 11:55	29.05.2015 11:55	
2	Monster <jagent@route.monster.com>	Neue Jobs (10): Meine Jobs per Mail	TEXT/HTML; charset=utf-8	29.05.2015 11:55	29.05.2015 11:55	
3	StepStone - Jobagent <info@jobagent.stepstone.de>	55 neue Stellenangebote bei StepStone	TEXT/HTML; charset=utf-8	29.05.2015 03:10	29.05.2015 03:10	
4	StepStone - Jobagent <info@jobagent.stepstone.de>	55 neue Stellenangebote bei StepStone	TEXT/HTML; charset=utf-8	29.05.2015 03:10	29.05.2015 03:10	
5	Monster <jagent@route.monster.com>	Neue Jobs (10): Meine Jobs per Mail	TEXT/HTML; charset=utf-8	28.05.2015 11:55	28.05.2015 11:55	
6	Monster <jagent@route.monster.com>	Neue Jobs (10): Meine Jobs per Mail	TEXT/HTML; charset=utf-8	28.05.2015 11:55	28.05.2015 11:55	
7	Monster <jagent@route.monster.com>	Neue Jobs (10): Meine Jobs per Mail	TEXT/HTML; charset=utf-8	28.05.2015 11:55	28.05.2015 11:55	
8	StepStone - Jobagent <info@jobagent.stepstone.de>	55 neue Stellenangebote bei StepStone	TEXT/HTML; charset=utf-8	28.05.2015 03:07	28.05.2015 03:07	
9	StepStone - Jobagent <info@jobagent.stepstone.de>	55 neue Stellenangebote bei StepStone	TEXT/HTML; charset=utf-8	28.05.2015 03:07	28.05.2015 03:07	
10	Monster <jagent@route.monster.com>	Neue Jobs (9): Meine Jobs per Mail	TEXT/HTML; charset=utf-8	27.05.2015 11:58	27.05.2015 11:58	

Abbildung 8: Ansicht E-Mails



#	Source	Received date	Value	Link	Downloaded
0	Monster <jagent@route.monster.com>	29.05.2015 11:55	SOFTWAREENTWICKLER (M/W) HYBRIS IN DRESDEN, BERLIN, JENA	Link	✔
1	Monster <jagent@route.monster.com>	29.05.2015 11:55	Softwareentwicklung Anwendungen/GUI (m/w)	Link	✔
2	Monster <jagent@route.monster.com>	29.05.2015 11:55	Perl Entwickler (m/w)	Link	✔
3	Monster <jagent@route.monster.com>	29.05.2015 11:55	Beschäftigte / Beschäftigter (Entwicklung und ABAP-Programmierung SAP SRM / MM)	Link	✔

Abbildung 9: Ansicht Links zu Arbeitsangeboten

Downloaded weist deutlich darauf hin, dass der Inhalt von Arbeitsangebot schon heruntergeladen und in Datenbank gespeichert wurde. Mit dem Delete-Icon kann der Link gelöscht werden.

Ansicht Arbeitsangeboten

Diese Ansicht ähnelt sich dem Link-Ansicht allerdings repräsentiert nicht die Links sondern heruntergeladene Arbeitsangebote, deren Inhalt auch mit Show-Icon angezeigt werden kann. Das Flag **Extracted** zeigt, ob die Metadaten des Arbeitsangebots extrahiert wurden.

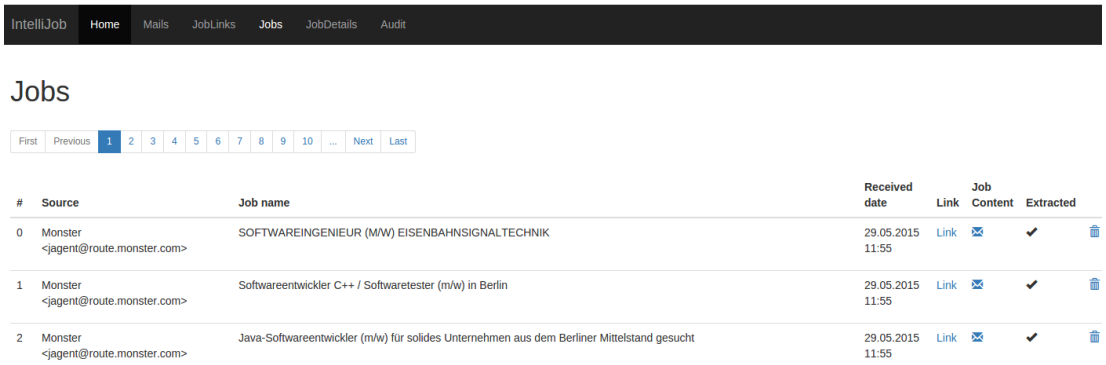


Abbildung 10: Ansicht Arbeitsangeboten

Ansicht Details zum Arbeitsangebot

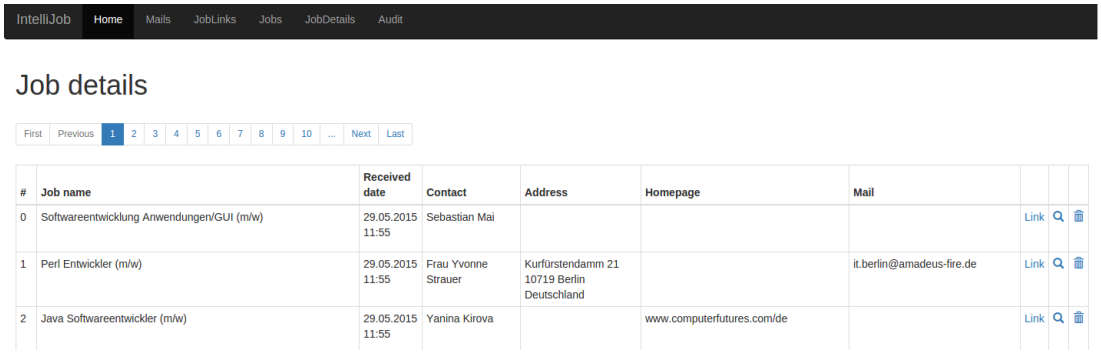


Abbildung 11: Ansicht Details zum Arbeitsangebot

Diese Ansicht zeigt die extrahierte Metadaten eines Arbeitsangebots wie **Berufsbezeichnung**, **Kontaktperson**, **Firmenadresse**, **Homepage**, und **Bewerbungsmail**. Einige Daten sind leer, weil sie entweder in dem Arbeitsangebot nicht gefunden oder nicht erfolgreich extrahiert werden konnten.

Ansicht Datenauswertung

Diese Ansicht wurde speziell für die Datenauswertung entwickelt und zeigt sowohl die aktuellste Ergebnis des Datenextraktions als auch die Historie über die früheren

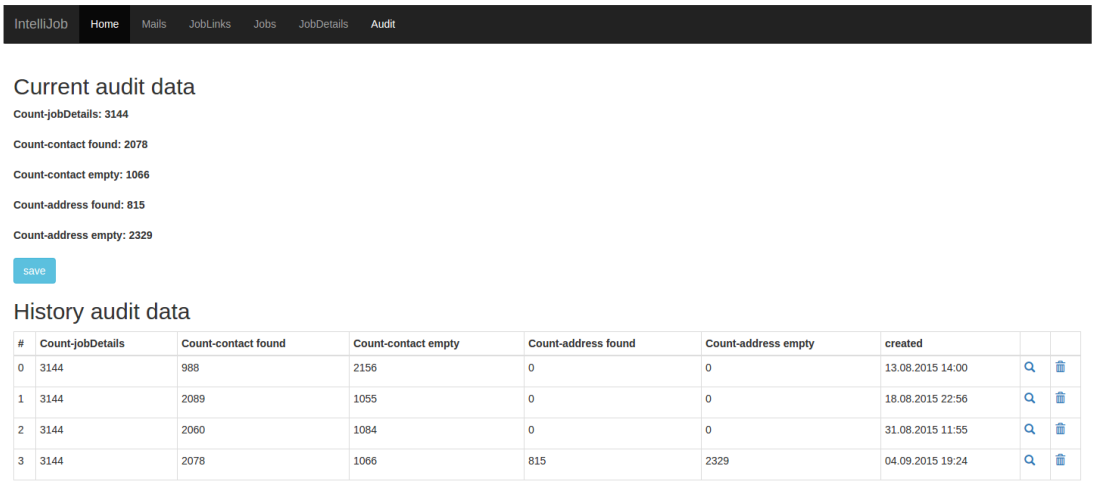


Abbildung 12: Ansicht Datenauswertung

Ergebnissen. Mit dem Save-Event wird die aktuellste Ergebnis in Datenbank gespeichert und zur Historie-Tabelle hinzugefügt. In der ersten Version werden nur folgende Daten ermittelt: **Anzahl aller Arbeitsangeboten**, **Anzahl aller Arbeitsangeboten mit gefundenen Daten-Kontaktperson**, **Anzahl aller Arbeitsangeboten mit fehlenden Daten-Kontaktperson**, **Anzahl aller Arbeitsangeboten mit gefundenen Daten-Firmenadresse** und **Anzahl aller Arbeitsangeboten mit fehlenden Daten-Firmenadresse**. Die gleiche Daten werden auch in der Historie-Tabelle erfasst.

5.3 Datenauswertung

Insgesamt gibt es 3144 Arbeitsangeboten und einige davon kommen doppelt vor. Dieses Problem ist bekannt und wird im nächsten Kapital noch detailliert beschrieben. Da das Address-Modell viel später entwickelt wurde, sind die ersten Historie-Einträge für dieses Modell null. Der erste Historie-Eintrag weist darauf hin, dass es nur 988 Kontaktpersonen von 3144 erkannt wurden. Es muss auch berücksichtigt werden, dass nicht alle Arbeitsangebote eine Kontaktperson, eine Mailadresse oder eine Firmenadresse haben. Außerdem widerspiegelt diese Zahl nur reine Logik von OpenNLP, das bedeutet Maxent-Algorithm mit Trainingsdaten. Denn die Sonderlogik, für den Fall, wenn OpenNLP keine Ergebnisse liefert, wurde zu diesem Zeitpunkt noch nicht entwickelt. Das erklärt auch den Sprung von 988 gefundenen Kontaktpersonen zu 2089 in dem zweiten Historie-Eintrag. Denn hier sind schon die Kontaktpersonen mit Hilfe von Namensdaten extrahiert worden. Bei der Analyse der Ergebnisse wurden Kontaktpersonen wie *Herr Berlin Deutschland* gefunden und der Grund dafür ist, dass der Vorname *Berlin* tatsächlich in der Namensdaten existiert. Deswegen muss das eigene Suchalgorithmus mit der Black-Liste von Namen erweitert werden. Die verbesserte Ergebnis ist in dem vierten Historie-Eintrag zu sehen. Der letzte Historie-Eintrag zeigt

alle Firmenadresse, die erkannt worden und auch die steigende Anzahl von gefundenen Kontaktpersonen. Der Grund für diese Steigung sind neue Trainingsdaten.

6 Problemen und Schwierigkeiten

Das größte Problem ist natürlich die Trainingsdaten. Täglich kommen neue Arbeitsangebote an. Die müssen zuerst analysiert werden, dann muss die richtige Stelle im Arbeitsangebot gefunden und markiert werden und letztendlich muss noch das Modell mit neuen Trainingsdaten trainiert werden. Dieser Prozess ist natürlich mühsam, sogar ein bisschen langweilig, muss aber gemacht werden, um bessere Ergebnismenge zu erzielen.
Das weitere Problem, das bis jetzt nicht gelöst werden könnte, ist doppelte Arbeitsangebote. Jedes Mail mit Arbeitsangeboten von Job-Portal beinhaltet neue Arbeitsangebote und die alte, die von gestern oder noch früher sind. Deswegen wird es vor dem Schreiben ins Datenbank geprüft, ob das Link zu diesem Arbeitsangebot schon früher gespeichert wurde. Durch die Beobachtung wurde erkannt, dass die Links zum gleichen Arbeitsangebot manchmal geändert werden. Außerdem kann ein Arbeitsangebot sowohl von StepStonde als auch von Monster Portal kommen. Die Links und Mail sind unterschiedlich und es gibt auch keine Möglichkeit die beiden Arbeitsangebote von einander zu unterscheiden, bis alle Daten erfolgreich extrahiert und mit einander verglichen werden. Wenn alle extrahierte Daten: Kontaktperson, Firmenadresse, Mail, Firmenhomepage, Berufsbezeichnung gleich sind, sind auch die Arbeitsangebote gleich.
Um ein Arbeitsangebot in Datenbank zu speichern, wird ein Link zu diesem Arbeitsangebot aufgerufen. Was ist, wenn das Angebot nicht mehr existiert? In dem Fall liefert Monster-Portal einfach HTTP-STATUS-404 (NOT FOUND) zurück, was natürlich richtig ist und von Softwareentwickler auch erwartet und gern gesehen wird. In der Realität kommt ziemlich oft vor, dass, was für Softwareentwickler gut ist, für Marketing nicht gut ist. Deswegen generiert StepStone-Portal eine Seite mit der Nachricht

> *„Diese Stellenanzeige ist nicht mehr verfügbar. Das gesuchte Stellenangebot ist leider nicht mehr verfügbar. Nutzen Sie unsere Suchfunktion oder vorgeschlagene Links auf dieser Seite um weitere für Sie interessante Jobangebote zu finden.“*

Dabei werden die Stellenanzeige vorgeschlagen, die nicht der gesuchten Arbeitskriterien entsprechen. Deswegen muss beim Speichern eines Arbeitsangebots von Stepstone auch geprüft werden, ob die Nachricht *„Diese Stellenanzeige ist nicht mehr verfügbar.“* vorkommt.

7 Zusammenfassung und Ausblick

In dieser Arbeit wurde anhand eines praktischen Beispiels gezeigt, wie bestimmte Daten aus einem Text extrahiert werden können, welche Probleme dabei auftreten können und wie und mit welchen Werkzeugen diese Probleme gelöst werden können. Außerdem wurde auch gezeigt, wie mit Hilfe von Apache OpenNLP Framewrok eigene Modele erstellt und erfolgreich eingesetzt werden können. Eine gute Ergebnis konnte mit dem eigenen Model ***Kontaktperson*** (ContactPersondFinder) erreicht werden. Insgesamt wurden folgende Daten erfolgreich extrahiert: Berufsbezeichnung, Mailadresse, Firmenhomepage, Kontaktperson und Firmenadresse.
Eine wichtige Funktion, die bis jetzt nicht realisiert werden könnte, ist Auslesung von Berufsanforderungen. Diese Daten werden benötigt, um die Suche nach persönliche Fähigkeiten und Kompetenzen zu implementieren. Dafür wird auch in der nächsten Version eine Web-Form entwickelt, wo der Arbeitssuchender seine eigene Fähigkeiten bewerten kann. Als Endergebnis werden dann die Arbeitsangebote nicht nur nach Datum, sondern auch nach persönliche Fähigkeiten sortiert. Die Arbeitsangebote, die am besten den persönlichen Fähigkeiten entsprechen, werden zuerst angezeigt.

Literatur

Wikipedia. *Informationsextraktion*. 2015. URL: https://de.wikipedia.org/wiki/Informationsextraktion.

Freiknecht, Jonas. „Big Data in der Praxis“. In: 2014.

OnPageWiki. *Natural Language Processing*. 2015. URL: https://de.onpage.org/wiki/Natural_Language_Processing.

Bazenov, Denis. *Classification with Maximum Entropy*. 2015. URL: http://bazhenov.me/blog/2013/04/23/maximum-entropy-classifier.html.

Reiter, J. „Bildverschärfung durch Lösung der Fredholmschen Integralgleichung 1. Art mittels der Maximum-Entropie-Methode mit astronomischen Anwendungen“. PhD thesis. Innsbruck, 1985.

Duller, Christine. „Die Maximum-Entropie-Methode zur Bestimmung von Mischanteilen“. In: (2003).